インターフェース増刊

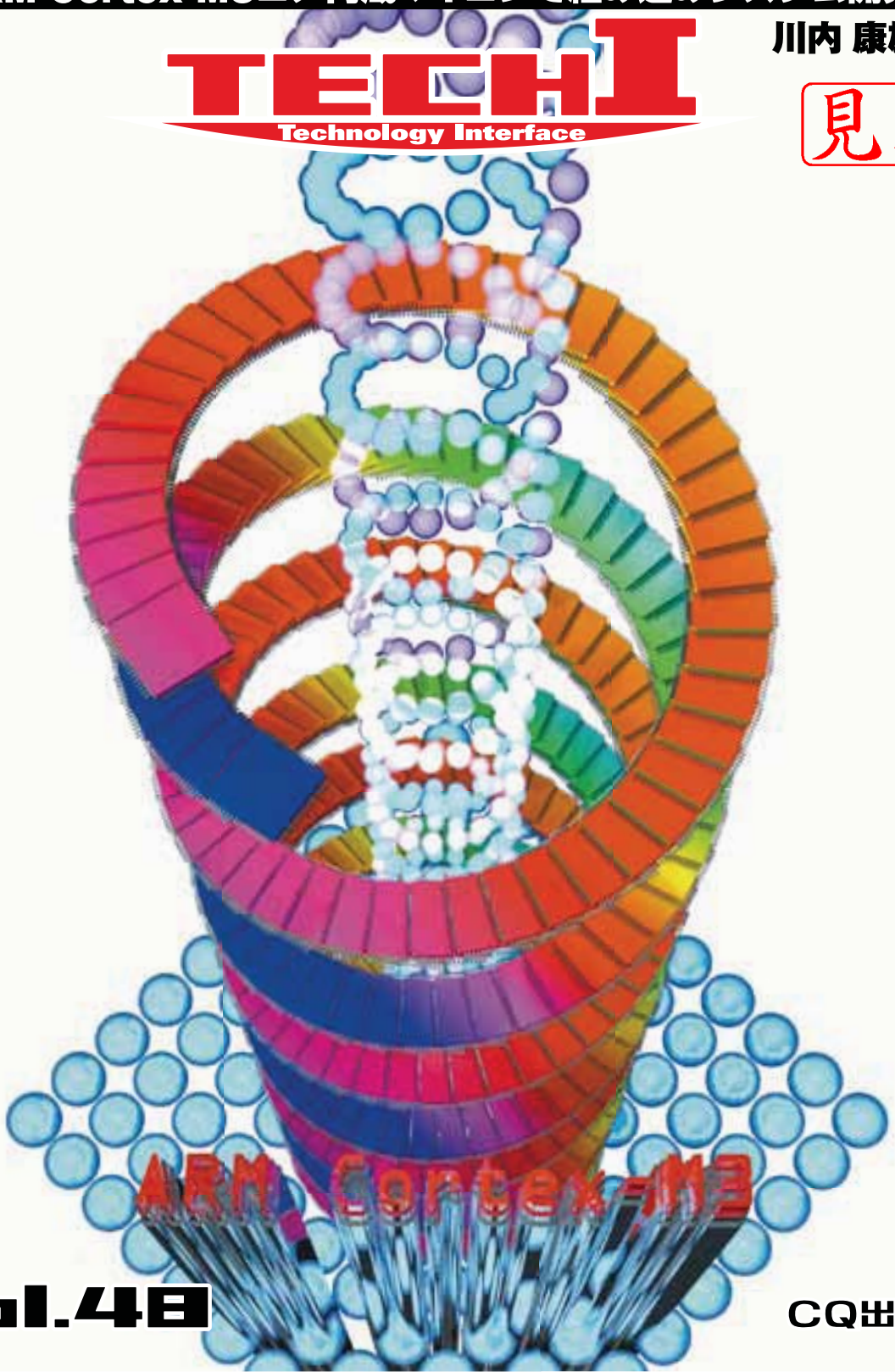
STM32マイコン徹底入門

ARM Cortex-M3コア内蔵マイコンで組み込みシステム開発

川内 康雄 著

TECH I
Technology Interface

見本



Vol.48

CQ出版社

第 1 章

インスパイアのための イントロダクション

1.1 マイコンで何ができるのか

あなたが本書を手に取った動機はさまざまでしょう. コンテストに出場するためのロボットを作りたい, 組み込みプログラミングを学んでキャリア・アップしたい, などなど…. 本書でマイコン開発を学ぶことで, あなたの目的をかなえてくれるはずです.

それでは早速マイコンとは…という話を始めたいところですが, 具体的なイメージで創作意欲をインスパイア(触発)したいと思います.

本書を読む動機が何であれ, マイコンを使うことは,

① プログラムで
② 制御される
③ モノを作る

ということが最終目的です.

作るという行為はとてもクリエイティブな要素を含みます. あれこれと考えながら思いを巡らすのはたいへんですが, とても楽しい作業です. しかし, 初めてマイコンに取り組むときに, ただ作れと言われても, 作りたいもの, 作れるものがいったい何なのか想像がつかないでしょう.

本書では, まずマイコンの「内蔵機能でできること」を「組み合わせ」ながら考えていきます.

マイコンでできることの一例を**表 1.1** に列挙します. **表 1.1** に示したのは, ①から③のうち, ②の制御する対象を意味します. なお, これは本書で紹介するマイコンの使用方法の例であり, マイコンでできることはこれだけに限りません.

これらの「できること」を見ると, なんだこれは(?)と思われた人もいるでしょう. こんなことはできて当たり前でしょうと. 確かにその通りです. LED を点灯させたり, 家電の電源を操作したりするだけであれば, わざわざマイコンを使う必要はありません. ホームセンターで売られている電球ソケット, AC 用ケーブル, スイッチ, コンセントをつなぎ合わせるだけです.

ただ, マイコンでの制御は「電球＋スイッチ」とは本質的に違う部分があります. それはプログラムで制御が可能であるということです.

「電球＋スイッチ」による制御であれば, 電球を点灯させるためには, 夜になったことを判断した人間がスイッチを押すという行為を行わなければなりません. 一方, マイコンによる制御であれば, マイコン内に内蔵したプログラムが, 意図する内容通りに, 電球を点灯させます.

電球＋スイッチ＋人間＝点灯の制御

を,

電球＋照度(明るさ)センサ＋マイコン＋プログラム＝点灯の制御

に置き換えることができたというわけです.

このような例であれば「マイコンで電球を点灯させることで何が便利になるんだ」と感じる人もいるでしょう. 確かにその通りです. それでは次の仕組みを作りたいときにはどうすればよいでしょうか.

「毎日夜になると点灯する電球. 夜とは午後 6 時から午前 6 時までをいう. ただし 6 月から 10 月までは午後 7 時から午前 5 時までを夜とする. 夜になっていなくても曇り空で暗いときには点灯する. 電球が切れて

表 1.1
マイコンでできること

スイッチを読み取る	LED を点灯させる	家電の電源を操作する
液晶に情報を表示する	モータを動かす	音を鳴らす
音声ファイルを再生する	障害物を検知する	方角を知る
ジョイスティックを使う	明るさを計測する	距離を計測する
無線通信する	傾きを計測する	情報を記憶する
時間を計測する	サーボ・モータを動かす	回転数を計測する

いて点灯しないときには，電子メールで通知する」

これも，「電球＋スイッチ＋人間」で実現することはできます．ただ，これを毎日間違いなく実行するとなると簡単ではありません．一方，この制御はマイコンで簡単に実現することができます．

電球＋照度センサ＋電流計＋リアルタイム・クロック＋ Ethernet コントローラ＋マイコン＋プログラム

で「電球＋スイッチ＋人間」と同じ，もしくは同等以上の制御を行うことができます．

ここで「マイコン＋プログラム」に着目してください．電球や各種のセンサは，それ自体では能動的な活動が行えない単機能の機器です．電球と時計はそれ自体ではお互いに何の関係もなく，独立して動作します．これを「マイコン＋プログラム」が中心となってつなぎ合わせることにより，新しい機能を生み出します．時計に連動した電球や，電流計に連動したインターネットも同様です．つまりマイコンは，「プログラムで機能を結びつける」ものだと考えることができます．

1.2 あなたならマイコンで何をしますか？

さてここで，プログラムを中心にして，さまざまな機能との組み合わせを考えてみましょう．自分が今プログラムができるかどうかは，気にする必要がありません．この際，回路や部品で本当にこんなことができるのかといったことも考えないようにします．およそ思いつくことは，だいたい実現できます．

たとえば，**表 1.1** を眺めていると，

ジョイスティック＋無線＋モータ＋プログラム＝ラジコン戦車

回転数計測＋方角計測＋情報記憶＋プログラム＝自転車用走行経路記憶機

サーボ・モータ＋障害物検知＋音声再生＋プログラム＝子供向け犬型ロボット

明るさ計測＋ LED ＋プログラム＝切り忘れ防止型トイレ照明

といった組み合わせを思いつきました．ここで**表 1.1** を眺めながら，10 個ぐらいは組み合わせを考えてみてください．まじめな実用品だけでなく，ジョークグッズ的なものも，まずは頭の中で作り出してください．組み合わせたものが実際に動作しているところを頭の中で想像すると，なかなか楽しいですよ．

1.3 大切なこと

さて，いろいろなアイデアは思いついたでしょうか．それでは早速製作を始めましょう．気後れすることはありません．大きなプロジェクトも結局は個々の小さな技術の組み合わせに過ぎません．

本書では**表 1.1** に記載した「できること」を実際に行うための回路の作り方，プログラムの書き方を紹介します．あとは実際に組み合わせたプログラムを作るだけで，プロジェクトは完成です．

もう一つ忘れてはいけない要素があります．それは好奇心と情熱です．「こんなことができるんだ」「自分もやってみたい」「自分ならこんなものが作りたい」「こんなものが欲しい」という想いがプロジェクトを成功

ブックガイド 1.1 C 言語入門

(1) 高橋麻奈；やさしい C，ソフトバンクパブリッシング，2001 年 9 月．

C 言語に初めて取り組む人向けに，「プログラミングとは何か」というレベルから解説しています．

(2) 三浦元ほか；組込み現場の「C」言語 基礎からわかる徹底入門 重点学習＋文法編，技術評論社，2010 年 1 月．

組み込み開発を想定した C 言語の入門書です．組み込み開発でよく用いるポインタや構造体の利用方法が詳しく解説されています．はじめて C 言語を学ぶという方にはちょっと難しいかもしれませんが，「やさしい C」を読んだあとであれば理解できるはずです．

(3) 矢沢久雄；プログラマの完全常識 開発者が知っておくべきプロの知恵，技術評論社，2007 年 1 月．

C 言語の本ではありませんが，プログラミングに入門する前か入門した直後あたりに読んでおくと，前提知識が整理できるのでよいでしょう．

(4) ハーバート・シルト；独習 C 改訂版，翔泳社，1999 年 2 月．

定番の入門書です．きちんと C を学んでみたいという人にお勧めです．

(5) Les Hancock ほか；改訂第 3 版 C 言語入門，アスキー出版局，1992 年 9 月．

コンパクトにまとまっていますが，本格的な C 言語入門書です．

第 2 章

マイクロコントローラのしくみ

2.1 マイコンの適用例

最近の電子炊飯ジャーの多くは，炊きあがり時間を設定したり，炊き方加減を選択したりできるようになっています．このような炊飯ジャーはマイコンの典型的な適用例です．

現代的な炊飯器であれば，「はじめちょろちょろ…」の火加減を炊飯器が調整し，ユーザが望んだ時間に炊きあがるという機能が実装(開発・製造の世界では一定の機能や性能，構造が備わっているときに「実装」されていると表現する)されています．この機能を実現しようとすると，物理的・電気的には，

- 釜の温度やふたが閉まっているかどうかを判別するセンサ
- 現在時刻と炊き始めからの時間を計測できるタイマ
- ユーザによる設定情報(炊きあがり時間・好みの炊き方)や炊飯器の現在状態を表示するための液晶
- 電熱器への通電や発熱量を操作するリレーやスイッチング素子
- ユーザからの操作を受け付けるボタン(スイッチ)

が必要になります．

これらの機能を電気回路で別々に作成することもできます．実際，マイコンが一般的でなかった時代には，ねじりバネや水晶振動子式のタイマと，サーミスタ(一定の温度で ON/OFF が切り替わるセンサ・スイッチ)を活用した「電気式」炊飯器が主流でした．このような回路を，物理的な配線(ワイヤ)により論理を実現しているという意味で，ワイヤード・ロジックと呼んでいます．

しかしこのような制御ではどうしても限界が出てきます．「火加減をだんだん強くする」や「毎朝 6 時 30 分に炊きあがる」といった制御は電気的・機械的制御では，不可能ではありませんが容易ではなく，また仮にできたとしても製作コストが高くなり，小型化が困難なものになってしまいます．またいったんワイヤード・ロジックとして回路・機械を完成させると，これを変更させるには，物理的な部分の再設計，再製作が必要になってしまいます．

そこでマイコンの登場となります．現在のマイコンには，センサ出力情報を把握するための A-D コンバータ，スイッチ入力を把握するための入力ポート，高精度な時間計測が可能なタイマ，液晶をコントロールする出力ポート，アナログ出力を可能にする D-A コンバータ/PWM コントローラがすべて内蔵されています．そのため，マイコンが 1 チップあれば，電子式炊飯器を構成する機能を結びつけるための中核的機能がすべて揃います．そしてこれらの機能の制御は，ワイヤード・ロジックではなく，マイコン内部に電子的情報として記録させるソフトウェアによって制御できます．ソフトウェアによる制御であれば，必要に応じて，随時，簡単に書き換えができます．このようなソフトウェアによる制御をプログラムド・ロジックと言います．

このような理由から家電製品の多くにはマイコンが使用され，便利な機能を実現しています．またマイコンの適用先は家電製品だけではありません．たとえば自動車では，エンジンの点火・噴射制御，変速制御，ABS(Antilock Brake System)，キーレス・エントリ，エアコンの温度制御，イモビライザなどのさまざまな制御にマイコンが使用されています．高級車では使用されているマイコンが 1 台あたり 100 個を超えているそうです．

2.2 なぜマイコンを使うのか

パソコンの世界でマイコンというと，昔のコンピュータやパソコン自体を指します．辞書では，マイコンとはマイクロコンピュータ(Microcomputer ：超

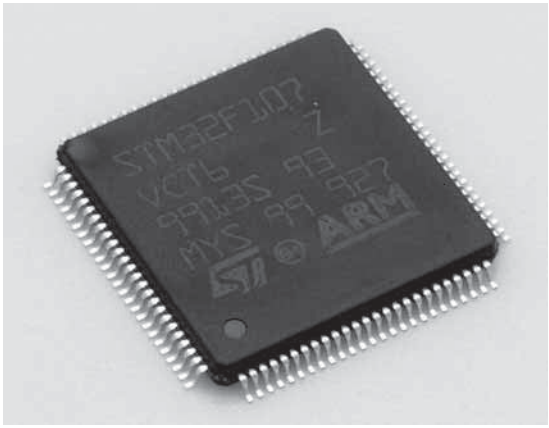


写真 2.1　マイクロコントローラ(STM32)

写真 2.2　パソコン用 CPU(Intel Pentium)

小型計算機)の略称と定義されています．これに対して本書で扱うマイコンはマイクロコントローラ(Microcontroller ：超小型制御器)の略称です(**写真 2.1**)．このマイコンも，超小型でかつ高度な計算機能を備えているので，超小型計算機であることに変わりありません．物理的な外形もほぼ同じです．ただ，計算機能に加えて，アナログ，ディジタルの入力機能，出力機能を備え，外部との物理的コミュニケーションが可能であり，外部機器の制御ができることから，コントローラ(制御器)と呼ばれています．

パソコンが使用している CPU(以下本書では「パソコン用 CPU」という．**写真 2.2** は Intel の Pentium)は高度化・高速化を続けており，最新型のパソコン用 CPU の計算能力は驚異的に高いです．そうすると，マイコンでなくても，パソコン用 CPU を使えば，高性能な「電子式炊飯器」を作ることができるのではないかと思えます．

しかし現実的にはそうはいきません．まずコストは，最新パソコン用 CPU はプロセッサ単体で数千円から数万円です．本書執筆時点で Intel の Core i7 920 プロセッサは CPU だけで 3 万円もします．しかし炊飯器の製品価格は高くても数万円です．炊飯器に使用される多くの部品の中で，CPU だけでこれだけの費用がかかってしまうと，炊飯器の製品価格が跳ね上がってしまいます．

もちろんこのような高機能なパソコン用 CPU を使用することによって，独自の機能・性能を備えられるということであれば，たとえ価格が高くても，製品としての存在意義はあるといえるでしょう．ただ，炊飯器であれば，1 秒間に 1GHz(秒速 10 億回)の計算を行えるからといって，炊きあがるご飯がおいしくなったり，操作が便利になったりということはありません．

一方，マイコンの価格は数十円のものもありますし，高くても 1,000 円程度です．このように低価格で入手できるマイコンであっても，炊飯器の機能を実現するために必要となる性能・機能は十分に備えています．

また，高性能な CPU ほど発熱量が多く，電力消費量が多くなります．一方，マイコンの消費電力は，パソコン用 CPU と比べればごくわずかです．エコロジー化で電力消費量の削減が叫ばれる中で，無駄に電力を消費することは許されないでしょうし，電気代が安く済むに越したことはありません．また炊飯器はコンセントにつないで使えますが，これが仮にバッテリで動作する機器であればどうでしょう．電力消費量と動作可能時間は反比例する関係にあります．携帯電話などで「長時間動作」が重要な性能指標であることは言うまでもありません．

そしてパソコン用 CPU には各種の入出力機能は内蔵されていないことがほとんどです．たとえば Core i7 プロセッサには，A-D コンバータや汎用入出力機能は内蔵されていません．外部との接続機能は，パソコン用 CPU の利用者側が接続機能自体を自ら選択し，パソコン用 CPU のバス(情報の伝送経路となる部分)に接続する形で増設していきます．そのため，パソコン用 CPU の利用方法や形態の自由度は広がります．しかし，これはパソコン用 CPU で外部機器を操作しようとすると，その周辺に別途コントローラ・チップや電子回路を実装する必要があるということです．そうすると基板のサイズがどんどん大きくなるので小型化が困難になります．別に部品を付けることになるので，コストも上がります．一方，マイコンには各種の電子的制御を行うために必要な周辺回路が最初から内

第 3 章

汎用入出力(GPIO)ポートの基本：ファームウェアを使用した初期化と操作

本章ではマイコンにおける基本周辺回路であるGPIOを解説します．またGPIOの使い方とあわせて，ファームウェア・ライブラリを使用したプログラミング方法も解説します．

3.1 GPIOの概要

GPIOは，マイコンが出力しようとする情報を電気信号のON/OFF(電圧の高低)という形で出力し，外部からマイコンに向けられた電気信号を，マイコンが情報として受け取るための周辺回路です．

このGPIOは実際にはどのような回路で構成されているのでしょうか．**図3.1**にSTM32のリファレンス・マニュアルの該当部分を示します．**図3.1**はGPIOの実際の回路図ではなく，GPIOの回路構成を抽象的に示した図です．点線で囲まれた上半分が入力回路，下半分が出力回路です．

3.1.1 出力として使う場合

図3.1の出力ドライバを見てください．真ん中あたりに「P-MOS」「N-MOS」という部品があります．MOSとはMOS FET(Field Effect Transister ；電界効果型トランジスタ)のことで，この各端子にはゲート(G)，ソース(S)，ドレイン(D)という名前が付いています．

MOS FETはトランジスタの一種ですが，ここではゲートに電圧がかかるかで，ソースとドレインの間に電気が流れるかどうかのコントロールができるスイッチだと考えておいてください(P-MOSはゲートにマイナスの電圧をかけたときに，ソースからドレインに電流が流れ，N-MOSはゲートにプラスの電圧をかけたときにドレインからソースに電流が流れる)．そしてV_{DD}はプラス側の電源，V_{SS}はマイナス側の電源を意味します．MOSの接点部分からは右側に線が延びており，I/Oピンにつながっています．このようにP-

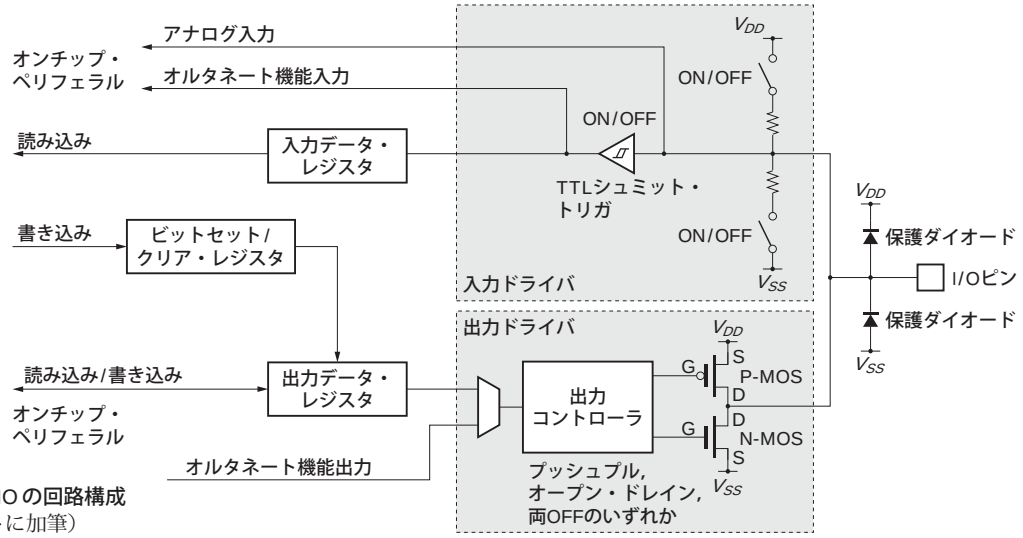


図3.1
STM32のGPIOの回路構成
(データシートに加筆)

「本文中のサンプル・プログラムは必要部分のみを抜粋したものです．プログラムの全体はwebページからダウン・ロードできます．開発環境の構築方法，プログラム・コードの確認方法，プログラムの実行方法はwebページに掲載の導入編を参照してください」

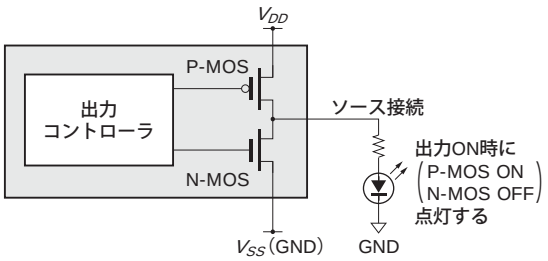


図 3.2　GPIO を出力として使用し，LED をシンク接続した場合の回路構成

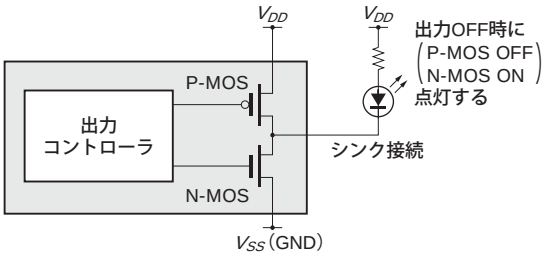


図 3.3　GPIO を出力として使用し，LED をソース接続した場合の回路構成

MOS と N-MOS が対称に構成されている回路を CMOS(Complementary MOS)と呼んでいます．

I/O ピンに発光ダイオードをつなぐと，**図 3.2** のような状態になります．出力と関係のない回路部分は省いています．

マイコンが出力を H(ハイ) にしているときには，上のスイッチ(P-MOS)は ON になって V_{DD} に接続され，下のスイッチ(N-MOS)は OFF となって切断されます．結果 I/O ピンに接続された LED は V_{DD} から V_{SS} に接続されているので，電流が流れて点灯します．

逆にマイコンが出力を L(ロー) にしているときには，P-MOS は OFF になって切断され，N-MOS は ON になって V_{SS} に接続されます．その結果 I/O ピンに接続された LED は，両端が V_{SS} に接続されていることになるので点灯せず，消灯したままとなります．

マイコンの出力は，I/O ピンをプラス側とマイナス側の電源につなぎ替えるスイッチです．ここで LED のつなぎ方を**図 3.3** のように変えてみましょう．

図 3.2 とは異なり，LED のアノード側が電源のプラス側に接続され，カソード側が電源のマイナス側に接続されています．このように接続したときに，LED は点灯するのでしょうか．

マイコンが出力を H にしているときには，P-MOS は ON になって V_{SS} に接続され，N-MOS は OFF になって切断されます．その結果，I/O ピンに接続された LED は，両端が V_{DD} に接続されているので点灯せず，消灯したままになります．

逆にマイコンが出力を L にしているときには，P-MOS は OFF になって切断され，N-MOS は ON になって V_{SS} に接続されます．その結果，I/O ピンに接続された LED は V_{DD} から V_{SS} に接続されているので，電流が流れて点灯します．**図 3.2** の回路ではマイコンの出力が H のときに点灯しましたが，今度の回路ではマイコンの出力が L のときに点灯します．

このように，P-MOS が ON になったときにマイコンから電流が流れ出る形で接続する方法をソース接続と言い，マイコンから流れ出ている電流をソース電流と言います．**図 3.3** の回路のように，電源のマイナス側のスイッチが ON になったときにマイコンに電流が流れ込む形で接続する方法をシンク接続と言い，マイコンに流れ込む電流をシンク電流と言います．

3.1.2　入力として使う場合

図 3.1 の入力ドライバにある「TTL シュミット・ト

コラム 3.1　ソース接続とシンク接続

マイコンの GPIO を出力として使い，LED のように電流を消費する部品を直接接続する場合には，通常はシンク接続を利用します．

GPIO 出力の内部回路は先述のように CMOS になっています．そのためソース接続でもシンク接続でも接続先の部品への電流の供給を制御できます．見た目の違いは，ON 時と OFF 時のどちらで電流が流れるのかと，接続先の部品を V_{DD} と V_{SS} のどちらに接続するのかという点です．

それでもシンク接続が望ましいのは，流せる電流量に違いがあるからです．後述のとおり MOSFET は，同じサイズであれば，P 型のものよりも N 型のものの方が多くの電流を流すことができます．N-MOS に電流を流すために使用できるのはシンク接続です．

そのためソース接続とシンク接続の許容電流量を分けて規定し，シンク電流の方が余裕があるマイコンが多くなっています．ただし本書では，「ON 時に ON になる」という分かりやすさを重視したのと，STM32 はソース接続でもシンク接続と同じ量の電流を流せるので，ソース接続を多く用いています．

第 4 章

汎用入出力（GPIO）ポートの活用：各種回路

ここまでのプログラムでマイコンの基本的な入出力が扱えるようになりました．本章からはLEDの点灯のような実験室的な回路だけではなく，実際の開発や製作で活用できる回路とプログラムの例を紹介します．

4.1 ソリッド・ステート・リレーによるスイッチ操作

4.1.1 ACコントロールとアイソレーション

マイコンから，AC100V動作の機器（要は家庭用コンセントから電源をとるもの）を操作してみましょう．ここではパソコンからの操作で扇風機をON/OFFさせてみます．

LEDのように直流で動作する素子であれば，たとえ必要な電流が大きくても，マイコンのGPIOに直接トランジスタを接続してドライブすれば動作させることができます．しかしトランジスタに流せる電流の方向は一定なので（NPNトランジスタであればベースとコレクタからエミッタに向けて流れる），そのままでは交流をコントロールできません．

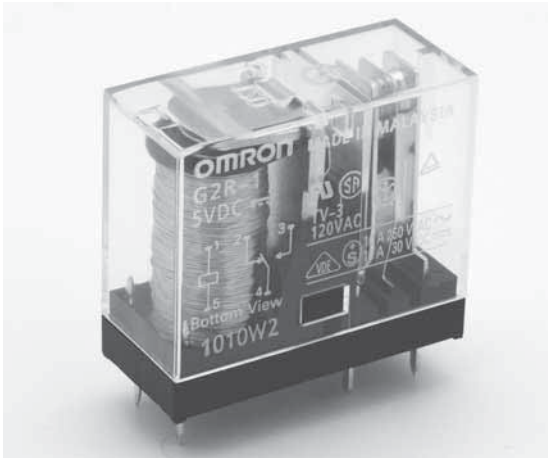


写真4.1　リレー（基板実装タイプ）

また扇風機のようにモータを使用している機器の場合，電源を切ったあと余勢で回っているときは発電しています．この電力がマイコン側に流れ込んでしまうと，マイコンの許容電圧や許容電流は大きくはないので，マイコンを傷めてしまう可能性があります．そのためON/OFFの信号の処置をする回路と，機器の動作に関わる回路とは絶縁されている必要があります．絶縁することを「アイソレーション（Isolation）する」，絶縁されている状態のことを「アイソレーションされている」と言います．

アイソレーションするためによく用いられているのは，リレーを使う方法です（**写真4.1**，**写真4.2**）．リレーは電磁石でスイッチを物理的に操作し，ON/OFFを行います．

電磁石を介して操作しているので，アイソレーションされており，スイッチ部分はごく単純な機械的な接点なので，直流/交流ともに流すことができます．

ただ，リレーは電磁石を使っているため消費電力が大きいことと，機械接点があることから寿命が長くはないというデメリットがあります．物理接点を用いて

写真4.2　リレーの接点の拡大写真（左半分が接点，右半分が電磁石 写真では右側の接点につながっている）

「本文中のサンプル・プログラムは必要部分のみを抜粋したものです．プログラムの全体はwebページからダウン・ロードできます．開発環境の構築方法，プログラム・コードの確認方法，プログラムの実行方法はwebページに掲載の導入編を参照してください」

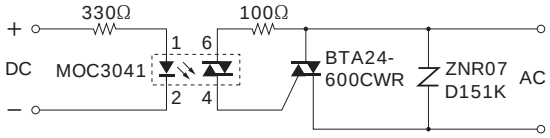


図 4.1　秋月電子通商のソリッド・ステート・リレー・キットの回路

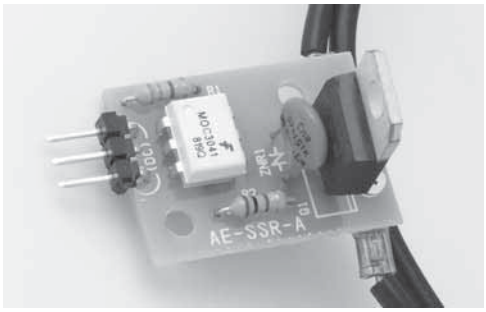


写真 4.3　電源ケーブルの途中部分を開く形で製作した

いるので，高速なスイッチングにも向いていません．そこで最近は半導体で構成されるソリッド・ステート・リレーがよく用いられるようになりました．なおソリッド・ステート・リレーだけではアイソレーションが難しいので，フォトカプラやフォト・トライアックを併用します．これらは光でアイソレーションを行います．入力側は LED を点灯させるようになっており，出力側は光に感応して ON/OFF が行われるフォト・ダイオードが内蔵されています．

ソリッド・ステート・リレーを利用する際には，自分で部品を集めて製作できますが，秋月電子通商から安価なキット〔ソリッド・ステート・リレー（SSR）・キット 25A（20A）タイプ 通販コード K-00203，250 円〕が発売されています．今回はこれを利用します．なお同社は製品の仕様変更や品番の改廃が多いので，実際に購入する際には同等品を選択するよう留意してください．

4.1.2　AC 機器操作の際の注意

AC 電源を利用する製作には特別の注意が必要です．

AC 電源の電圧は 100V です．100V が流れている回路に手が触れると感電し，場合によっては重篤な障害を負うことになります．回路のどの部分に 100V が流れているのかを正確に把握し，使用するときだけコンセントに接続するようにしてください．

本書の例ではキットの基板がむき出しとなっていますが，製作者以外が使用することがある機器の場合には，かならずケースなどに格納し，誤って回路に触れることがないようにしなければなりません．

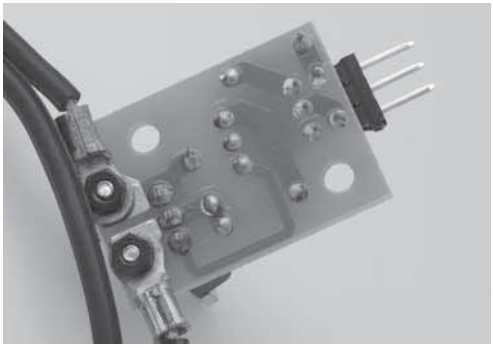


写真 4.4　裏面からターミナル端子で固定した

感電しないにしても，AC100V が 100V 用でない回路に供給されてしまうと，瞬時にその回路を破壊してしまいます．回路の実験前には，100V 系の回路とそのほかの回路がショートしていないか，テスタでチェックしてください．また AC 電源は電圧が高く，供給される電力量も潤沢なので，ショートさせてしまうと簡単に発火します．発火する前にブレーカが落ちるケースもありますが，保証はできません，電気設備を痛めてしまう可能性もあります．プラグの端子間も確実に絶縁されているか，慎重にチェックしてください．

電力量や電流量の許容範囲を超えることから，AC 100V が流れる回路をブレッド・ボード上で組んではいけません．

また詳細な解説は省きますが，電気設備の一部となる部分，たとえばブレーカとコンセントの間にソリッド・ステート・リレーを設置するには電気工事士の資格が必要です．加えて AC 電源に接続される機器を開発して商用製品として販売する際には，PSE マークを取得する必要があります．安全確保のための規制なので，かならず遵守するようにしてください．

4.1.3　キットの製作

キットの製作自体は説明書通りに行えば，とくに問題はありません．回路を**図 4.1** に示します．筆者は，延長コードの間にキットを挟み込む形で製作しました（**写真 4.3**）．

細かい工夫としては，①入力端子部分はマイコンとの接続を考慮してヘッダ・ピンとし，②出力部分はソリッド・ステート・リレーに付けるヒートシンク（放熱板）と干渉しないように，ターミナル用端子を圧着したケーブルをねじ止めしました（**写真 4.4**）．

第5章

システム・タイマ：タイマと割り込みの基本

システム・タイマはCortex-M3コアが備えているシンプルなタイマです．簡単に利用できるので，タイマと割り込みの基礎を，システム・タイマを題材にして解説します．

なおシステム・タイマはCortex-M3コア自体にもともと備わっている機能です．そのためリファレンス・マニュアルには解説がありません．レジスタの内容などの詳細を確認したい場合には，「Cortex-M3テクニカルリファレンスマニュアル」の中の「8.2.2 NVICレジスタのマップ」を参照してください．

5.1 システム・タイマと割り込みの連携動作

システム・タイマは24ビットのカウンタでできています．マイコンの世界では「8ビットタイマ」といえば，8ビットのビット長の最大の数字まで数えることができるカウンタを意味します．24ビットの最大値である0b 11111111 11111111 11111111は10進数で16,777,216ですから，およそ1600万まで数えることができます．これだけ数えられるのであれば，任意の時間をシステム・タイマだけで計測できそうにも思えます．しかし，システム・タイマはシステム・クロックが発生するたびにカウントされます〔正確にはAHBクロック(HCLK)．詳しくは第2章参照〕．つまり最大で72MHzもの速さでカウントされます．1秒間に7200万カウントするため，システム・タイマのカウントを最大限に使っても，0.2秒程度になります．そのため時間などをカウントする際にはプログラムでの工夫が必要です．

システム・タイマはダウン・カウント・タイマです．あらかじめタイマに値を設定し，クロックなどが発生するたびにカウンタが1ずつ減っていきます．システム・タイマではカウンタの値がゼロになると，割り込みが発生します．その後カウンタは自動的に前の値に戻され，またダウン・カウントを開始します(**図5.1**)．ちなみに，0からスタートして，設定した値までカウンタを1ずつ増やしていくタイマをアップ・カウント・タイマと言います．

割り込みはマイコンが現在行っている処理を強制的に中断させて，ほかの処理を行います．システム・タイマの割り込みを利用すると，一定時間が経過するごとに行いたい処理を実行できます．たとえば，システム・タイマ割り込みが発生するたびごとにLEDの点灯/消灯を切り替えれば，マイコンが別のプログラムを実行している間でも，LEDを点滅させることができます．

時間の計測であれば，たとえばシステム・タイマ割り込みを1msごとに発生させて，そのたびにカウント用の変数の値を1ずつ増やしていけば，カウント用変数が1000になった時点で，1秒が経過したと判断できます．次項で紹介するサンプル・プログラムの`Delay`関数では，割り込みとシステム・タイマを組み合わせています．`Delay`関数は，引数として渡されたミリ秒の間，何もしないで待つという関数です．ここではシステム・タイマは「1ミリ秒が経過するごとに割り込みを発生させる」ように設定しています．そして割り込み処理には，「システム・タイマ割り込みが発生したらカウンタ変数を1減算する」という処理が登録されています．`Delay`関数は常時カウンタ変数が0になっていないかを監視しており，割り込みによってカウンタ数が0になった時点で無限ループから抜けて，関数の呼び出し元に処理が戻るようになっています．

「本文中のサンプル・プログラムは必要部分のみを抜粋したものです．プログラムの全体はwebページからダウン・ロードできます．開発環境の構築方法，プログラム・コードの確認方法，プログラムの実行方法はwebページに掲載の導入編を参照してください」

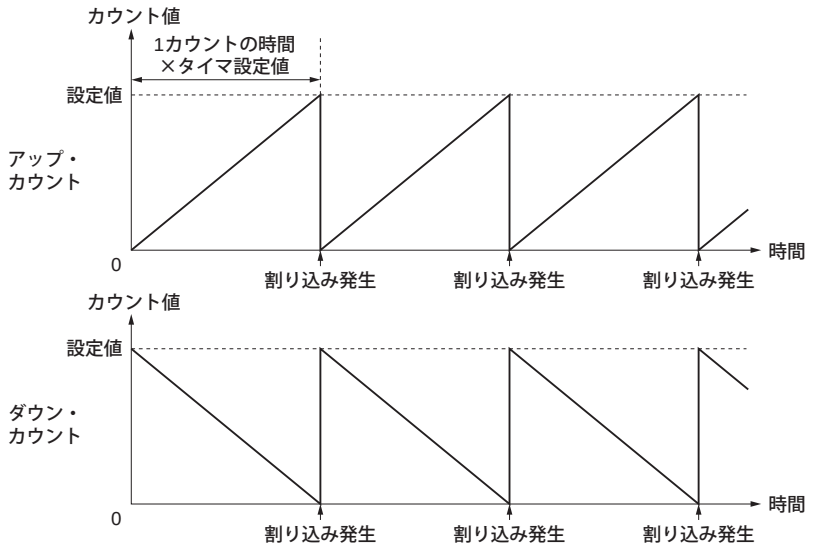


図 5.1
アップ・カウントと
ダウン・カウント時
のカウンタ値の変化

ライブラリ 5.1　SysTick_Config 関数

関数プロトタイプ	`static __INLINE uint32_t SysTick_Config(uint32_t ticks)`	
動作	システム・タイマの①割り込み有効化，②カウンタ初期化，③再ロード・レジスタの設定，④スタートを行う	
引数	`ticks`	再ロード・レジスタの値を指定する．システム・タイマはこの値から 0 までカウントする
戻り値	正常に実行を終了した場合 0 を返す．不正な ticks（システム・タイマのカウント上限値を超える値）が与えられた場合 1 を返す	

.2　LED の点滅処理

5.2.1　割り込みによる直接操作

サンプル・プログラム	systick_ob_led_toggle_by_interrupt_only

ここではコードを見ながら解説していきましょう．サンプル・プログラムは 1 秒ごとにオンボード LED を点滅させます．「1 秒ぐらい」や「たぶん 1 秒」ではなく，本当の 1 秒であるというのがポイントです．これまでのサンプル・プログラムでは，`main.c` だけを使用していましたが，ここからは `stm32f10x_it.c` や `stm32f10x_it.c`，そして `main.h` も使用します．マイコン・ボードに回路を接続する必要はありません．

まずは `SysTick_Config` 関数（**ライブラリ 5.1**）を使用して，システム・タイマを初期化してスタートさせます．なお `SysTick_Config` 関数は CMSIS の一部であり，`core_cm3.h` 内でインライン関数として定義されています．CMSIS は同じ Cortex コアを内蔵するマイコン間でアプリケーションの相互互換性を高めるための規格や仕様，フレームワークの総称で，ARM が定めています．

STM32 では，アップ・カウント・タイマの「この値までカウントしたらカウンタを 0 に戻す値」，ダウン・カウント・タイマの「最初にタイマに格納されている値：カウンタが 0 になったときに，再びカウンタに設定する値」を再ロード・レジスタや自動再ロード・レジスタと呼んでいます．

システム・タイマは 0 になったときに割り込みが発生するのではなく，0 になってからその次のカウントで割り込みが発生します．そのため自動再ロード・レジスタに設定する値は「カウントしたいカウント数－1」となります．`SysTick_Config` 関数の場合，内部の実装で渡された引数 `ticks` の値から 1 を引いた数を自動再ロード・レジスタに設定しています．そのため `SysTick_Config` 関数を使用する場合は，カウントしたい値をそのまま引数として渡せばよいことになります．

サンプル・プログラムでは `main.c` の**リスト 5.1** で `SysTick_Config` 関数を使用しています．`SystemCoreClock` は `system_stm32f10x.c` で定義されている定数で，システム・クロックの周波数となっています．72MHz であれば 72,000,000 です．

システム・タイマはシステム・クロックと同じ速度でカウントしているので，たとえば再ロード・レジスタに 72,000,000 を設定すれば，1 秒をカウントできま

見本

このPDFは，CQ出版社発売の「STM32マイコン徹底入門」の一部見本です．

内容・購入方法などにつきましては以下のホームページをご覧下さい．
内容　http://shop.cqpub.co.jp/hanbai//books/49/49861.htm
購入方法　http://www.cqpub.co.jp/hanbai/order/order.htm

雑誌　01620-1
Ⓛ-2011.2/1

4910016200119
02200

インターフェース増刊

CQ出版社
〒170-8461 東京都豊島区巣鴨1-14-2
TEL.(03)5395-2141(販売部)

定価2,310円 本体2,200円